Genetic Algorithm Approach to Solving Sudoku

Rafael Galiev
ČVUT–FIT
galieraf@fit.cvut.cz

April 23, 2024

1 Introduction

Sudoku is a logic-based puzzle consisting of a 9x9 grid divided into 3x3 subgrids. Each row, column, and subgrid must contain all digits from 1 to 9 without repetition. Genetic algorithms, inspired by natural selection, apply this principle to problem-solving, evolving a population of potential solutions over generations. They are well-suited for complex problems like Sudoku, where traditional algorithms may struggle with the vast search space.

2 Implemented Algorithm

A genetic algorithm with individuals represented as a list of lists, each inner list corresponding to a Sudoku block, was used to evolve solutions. Two types of crossover mechanisms—single-point and two-point—were implemented to allow the exchange of blocks between individuals. To facilitate the evolution towards a solution, the algorithm utilized a roulette wheel selection mechanism. This method favored individuals with superior fitness scores, enhancing their chances for selection and subsequent reproduction. Mutation was carefully applied to avoid violating Sudoku rules, preserving the integrity of the blocks.

3 Common problems

To maintain the uniqueness of numbers in each row, column, and block during genetic operations, which posed a significant challenge, especially during crossover and mutation phases, initial population blocks were generated with unique values. Crossover operations were conducted at the block level to preserve this uniqueness within each block.

When the genetic algorithm risked convergence on local optima without further improvements, a restart mechanism was implemented. If no improvement was observed after a predetermined number of generations, the entire population was replaced with a newly randomized one. This method injected fresh diversity into the population and steered the algorithm back on course towards finding the solution.

4 Results

The genetic algorithm's performance varied with the mutation rate and the population size. The algorithm managed to solve puzzles with a moderate number of missing values effectively. However, as the difficulty increased with more missing values, the performance tended to fluctuate, indicating a need for further parameter optimization. The table above provides a detailed analysis of the genetic algorithm's performance across various configurations of missing values and parameters. The average execution times listed were computed over 5 runs of the algorithm. Notably, the execution time for 59 missing values is faster than for 57 missing values.

Table 1: Genetic Algorithm Performance Analysis.

Missing Values	**Population Size**	**Mutation Rate**	**Average Time**
10	2	0.10	0.729s
15	20	0.10	0.1000s
25	30	0.10	0.4850s
35	50	0.10	1.185s
43	5000	0.10	6.282s
49	5000	0.10	11.886s
57	2000	0.10	4m 6.1s
59	5000	0.10	5m 11s

5 Conclusion

The exploration of genetic algorithms for solving Sudoku puzzles has demonstrated that, although capable, they may not be the most efficient or preferred method in practical scenarios. Traditional approaches like backtracking, constraint propagation, and even more sophisticated methods like Dancing Links (Algorithm X) often provide more reliable and faster solutions.